Performance Modeling for the High Level Architecture

Sudhir Srinivasan
Mystech Associates, Inc.
5205 Leesburg Pike, Suite 1200
Falls Church, VA 22041
sudhirs@mystech.com

Paul F. Reynolds, Jr.
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
reynolds@Virginia.EDU

KEYWORDS
HLA, performance analysis, simulation modeling, federation analysis tool

ABSTRACT

Success of the High Level Architecture (HLA) is determined in large part by the performance of federations of HLA-compliant simulations. This paper describes an on-going performance modeling effort sponsored by DMSO to construct a configurable simulation model of HLA-compliant federations in order to obtain first-order performance characteristics of current and future federations. Supplementing the prototype experiments, this *federation analysis tool* will provide an economical means for a broader range of non-intrusive tests. It is expected that future federation designers will use this tool to verify the feasibility of their federation designs and to identify potential design refinements.

The infrastructural nature of the HLA poses some unique requirements and challenges in designing a simulation model. The HLA is expected to support a wide variety of simulations: Platform, Aggregate, Engineering, Analysis, time-stepped, event-driven, coordinated, independent, real-time, as-fast-as-possible, etc. These different types of simulations typically also have different performance requirements: communication latency, response time, efficiency of synchronization, repeatability, etc. Further, a wide range of hardware and software environments is expected. The challenge is to construct a single simulation model flexible enough to capture this wide variety of federation characteristics. This paper describes the modeling approach we have taken to meet this challenge.

1. INTRODUCTION

As a result of the recent DoD requirement for a common technical framework to facilitate interoperability and reuse among heterogeneous simulations (Objective 1 in [DoD95]), the Architecture Management Group (AMG) of the Defense Modeling and Simulation Office (DMSO) has undertaken the design and development of a *high level architecture (HLA)*. The HLA is one of the three parts of the common technical framework (the other two being the Conceptual Models of the Mission Space and Data Standards), enabling interoperability by requiring the specification of common semantics for simulations and the interactions among them.

The HLA consists of three parts:

- the *object model template (OMT)*, used to define the object models for individual simulations (*simulation object models* or SOM's) and for *federations* of simulations (*federation object models* or FOM's),
- the *interface specification*, describing the interfaces used by objects to interoperate with each other, and
- a set of *rules*, which encapsulate the guiding principles behind the HLA and the way in which it is intended to be used.

The services in the interface specification are implemented in the form of a *run-time infrastructure* (RTI) which provides the mechanism for interoperability among the simulations.

The Modeling and Simulation Master Plan [DoD95] states that by the second quarter of FY97, all ongoing DoD modeling and simulation programs will be reviewed for immediate HLA-compliance. Since HLA-compliance may affect the way in which simulations interoperate, performance of federations designed and built in conformance with the HLA is critical. The HLA effort should not hinder performance requirements in pursuit of the goal of making simulations interoperate successfully.

In addition to prototyping efforts, DMSO has sponsored the construction of a configurable simulation model to facilitate performance analysis of HLA-compliant federations. This paper describes the design of the simulation model, highlighting the unique challenges faced due to the wide spectrum of simulations and corresponding performance requirements encountered in the HLA context.

2. THE TASK

Our task is to design and develop a *federation analysis tool* that will capture the relevant performance characteristics of HLA-compliant federations (including the RTI implementation chosen), allowing federation designers to model and study their federations prior to constructing them. The tool focuses on the performance aspects rather than functional aspects of federations, providing first-order performance analysis capabilities that can be used by designers of future federations to identify potential design problems and refinements.

Modeling and Simulation is used within the DoD for a variety of reasons including training, analysis, acquisition and design. Since the HLA is intended to be the underlying architecture for *all* DoD models and simulations, it must meet a wide range of requirements. The AMG has identified a representative set of simulations and grouped them into four prototype federations, or *protofederations*, which are being made HLA-compliant as demonstrations of the HLA-concept. The spectrum of simulation types and performance requirements covered by these protofederations is described in Table 1.

The wide variety of simulation types presents a challenge in the design of a single simulation model: the model should be flexible enough to capture the different simulation types with ease and yet provide sufficient detail to produce useful results. Further complicating the task is the fact that a wide range of hardware/software environments is also expected (for example, communications could be Ethernet, ATM, DSI, ScramNet, Myrinet, etc.). Our approach is to base the design of the model on the current protofederations based on the premise that they are representative of future federations. Since we focus more on the characteristics of the individual simulations rather than the protofederations themselves, we believe the model can be easily extended to accommodate most future federations. Clearly, new concepts may emerge in future federations that are not represented in the protofederations of today. These will require appropriate enhancements to the model.

Name	Description	Federates	Purpose	Features	Requirements
Platform Protofederation (PPF)	Platform-level DIS-based	BFTT JTCTS BDS-D CCTT	Training of soldiers and staff	Real-time Independent Time-stepped	Low latency communications Low overheads
Joint Training Federation (JTFp)	Constructive ALSP based	Eagle NASM/AP NSS DEEM JTF HQ	Training of commanders	Loose real-time Coordinated Event stepped	Low overhead synchronization
Analysis Protofederation (AP)	Constructive	JWARS MIDAS	Analysis	Coordinated As-fast-as-possible Time/event stepped	Repeatability Low overhead synchronization
Engineering Protofederation (EnggPF)	Subsystem level High fidelity	AFEWES ACETEF REDCAP IADS SBD JMASS	Design Acquisition	Real-time Independent Time/event stepped	Repeatability Throughput Low latency communications

Table 1 - Protofederations

3. SIMULATION MODEL DESIGN

Our goal is to construct a simulation model with sufficient flexibility (i.e. it should be adequately parameterized) so as to allow users to model foreseeable HLA-compliant federations with ease. Since this is a performance model rather than a functional model, it is a stochastic abstraction of an HLA federation, focusing on resource usage and contention and ignoring the application-level details of the simulated federation. Rather than simulating the various application-level details of a federation (and thus effectively re-implementing the federation), we represent them by stochastic phenomena. However, those aspects of a federation that concern relevant performance metrics (such as resource usage) are modeled in detail. For example, while the generation of a message may be based on a stochastic representation of a federate, the transportation of the message from sender to receiver is modeled explicitly.

Figure 1 shows the logical and physical views of a typical HLA federation (we have omitted the layer commonly called "middleware" since it is not relevant to the performance model). While the RTI appears logically as a layer above the underlying communications of the federation, physically, the RTI is implemented as a combination of distributed components located at each federate as well as a centralized component (which could be absent in some RTI designs). Based on this figure, our model consists of two submodels: a *federate submodel* (including the local RTI component) and a *communications submodel.* A separate model of the central RTI component is not required since it can be simulated using the federate submodel. The communications model encapsulates *all* physical communications in the federation.

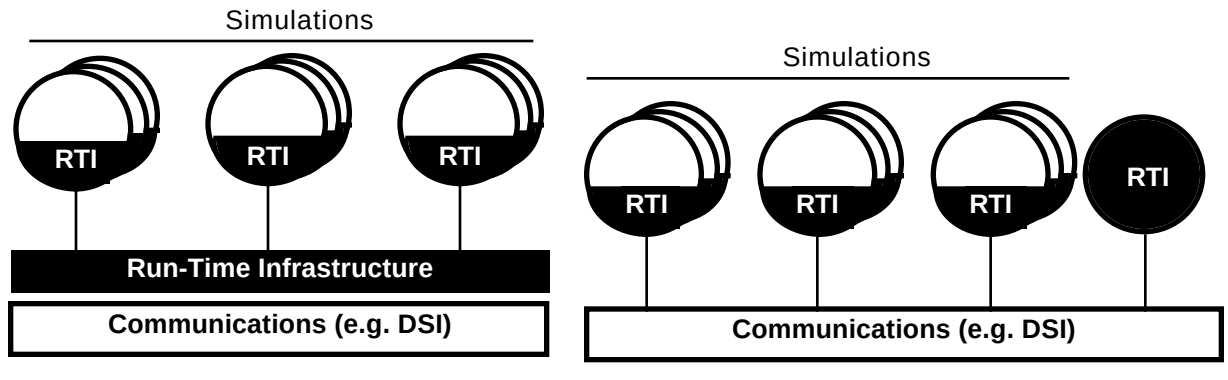
3.1 Federate Submodel

At the time of writing, the physical definition of a federate is still nebulous - the HLA Glossary [DMSO96a] defines a federate only as a member of an HLA Federation. For the present, we consider a federate as a simulation residing on a single *machine*, with a single interface to the RTI. Since we do not focus on the semantics of federates, we can model federates that span multiple machines easily by considering a federate as a collection of machines (although the issue of whether the federate has multiple interfaces to the RTI is still open and will be considered as the model evolves).

Given the goal of building a general model for a wide variety of federations, we require a general abstraction of a federate that is flexible enough to capture many types of simulations. The focus of the submodel should be on resource usage and contention. Thus, the federate submodel has three components:

- the workload characterizing the federate simulation, which constitutes the major source of resource usage,
- the local RTI component, which will place additional load on the machine resources and is therefore of interest, and
- a *system* component that models relevant low-level activities in the machine (such as sending and receiving of messages), which can become an important source of resource utilization depending on the implementation of the RTI.

The latter two of these will be identical across all federates and can thus be simulated in detail. What remains is a general characterization of a federate. Such a characterization is in fact provided by the HLA in the form of the *object model.* All HLA-compliant federates and federations must be represented using an object model. A *simulation object model* (SOM) describes a particular simulation while a *federation object model* (FOM) describes an entire federation. Both of these object models



(a) Logical View (b) Physical View

Figure 1 : HLA Federation

are presented using the object model template - OMT, which includes the class structure, component structure, attributes of objects, interaction structure among objects and associations. This object-based representation of federates provides an appropriate and convenient abstraction for use in our model.

Figure 2 shows the logical structure of a federate submodel. The first component is SIM, which consists of a collection of simulation objects representing the simulation. Objects can be public (solid) or private (shaded) depending on whether they interact with other objects outside the federate. Public objects generate RTI calls that are passed on to the RTI component for processing and routing.

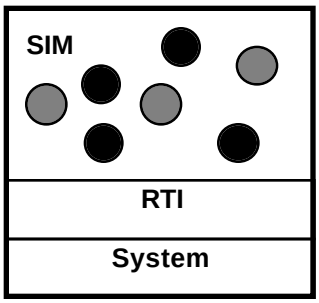


Figure 2 - Federate submodel

Processing may entail book-keeping operations that maintain the routing information as well as generation of system calls which are passed on to the System component. The System component simulates the actions of sending/receiving messages to/from the communications submodel. We describe each of the three components next.

3.1.1 SIM component

This component models the actual execution of events by the federate. As such, it must be capable of simulating both time-stepped and discrete-event federates. It may be noted that fundamentally, these two paradigms are equivalent, with the only practical difference being in the way simulated time is advanced - in the former, it is advanced once per timestep (which may involve multiple event executions) and in the latter, it is usually advanced after each event execution. We do not model individual events explicitly; rather, the execution of an event is considered the same as a single simulation or execution of the object that is responsible for that event. Thus, the sequence of event executions translates to a sequence of object executions. A fundamental design choice we have made is that the majority of the characterization of the federate is placed in the description of the objects. Thus, most of the stochastics in a federate are stored in the objects and the SIM component provides the dynamics for object execution. A more detailed discussion on objects and the data associated with them is deferred to Section 3.2

The SIM component is a loop, in each iteration of which it executes the following actions:

- Examine each object for execution and execute it if the data stored in the object so dictates. Execution of the object usually consists of consuming machine resources (such as CPU) and generating RTI calls.
- Advance simulation time of the federate. For time-stepped simulations, this would be performed once in each iteration of the SIM component whereas for discrete-event simulations, it would be performed after each object execution. Thus, a single model accommodates both types of simulations. As noted in Section 3.4, this and other time management aspects of a federation will be incorporated into future versions of the model.
- Perform overhead tasks such as computing line-of-sight and dead-reckoning remote entities. Currently, these tasks are performed once per iteration of the SIM loop; in the near future, the model will include the capability to specify these overheads for each object execution.

3.1.2 RTI Component

RTI calls made by executing objects in the SIM component are handled by the RTI component. This component has two main responsibilities:

- maintaining the data structures that determine where updates and interactions should go, and
- generating the messages and the lists of destinations for updates and interactions.

Note, we take the approach that the sender of a message determines the list of destinations. However, this does not imply that the actual implementation of the RTI is also sender-based (in fact, it is very likely it will *not* be sender based). The point is that the functionality is concentrated at the sender, *but not necessarily the costs*. We have separated the functionality from the costs by adopting a separate *cost model* for the RTI (explained in Section 3.3). This provides the flexibility of capturing various RTI implementations simply by adjusting the costs in the RTI cost model appropriately - the functionality is fixed (and located at the sender for convenience).

The RTI component maintains a complex set of data structures to track the publish/subscribe information among federates and uses this information to determine where updates and interactions should go. This determination is done stochastically, based again on parameters stored in objects invoking the RTI calls. At the sender, the RTI generates system calls that are passed to the System component. At the destination, the RTI component receives messages from the System component and forwards them to the SIM component after due processing. As noted in Section 3.3, the costs in the RTI

cost model are implemented by consuming machine resources.

3.1.3 System Component

The System component acts primarily as a conduit for messages, incurring appropriate costs as messages are sent and received. This component was included since these low-level costs can dominate performance depending on the implementations of the federates and the RTI.

3.2 Objects

As noted earlier, a federate is essentially determined by the objects it is comprised of. Objects contain no application specific data (such as attributes of a tank), but rather contain stochastic parameters that determine their dynamic behavior. Parameters for an object include the following:

1. The class to which it belongs, which determines the default set of classes to which this object is subscribed[†].
2. Mean time to execute the object once, which determines the CPU time requirement for executions of the object.
3. Current activity state, transition matrix and selection vector. To model the spectrum of objects encountered in typical simulations, we have introduced the concept of the *activity state* of an object. An object can be in one of three states of activity: *high*, *medium* or *quiescent*. The activity state determines primarily, the probability that the object will be executed at any time. Thus, each object has a *selection vector* consisting of three probabilities, one for each activity state. It is expected that P[*high*] will be greater than P[*medium*], which will be greater than P[*quiescent*], which will be nearly zero. Note, these probabilities directly affect resource utilization. The selection vector is used by the SIM component in deciding whether to execute a particular object in any given iteration of its loop. The *state transition matrix* controls the movement of an object from one activity state to another. This is a 3x3 matrix of probabilities as shown in Table 2. The probability in any cell is the probability that the object will move to the column-state given that it is in the row-state. It can be verified that the matrix in Table 2 describes an object that tends to be highly active but that can become quiescent occasionally. The state transition matrix is used by the SIM component each time it examines an object for possible execution. Irrespective of whether the object is executed in that particular iteration, its state may be changed according to the matrix.
4. The RTI calls generated by an object are determined by a combination of three parameters. First, a pair of Boolean flags determines whether this object is capable of generating messages to the external world or not (i.e. whether it is public). Two flags are required since an object can generate attribute updates and interactions independently of each other. The second parameter is an *RTI call vector* which is a vector of probabilities for various RTI call types. Each element of this vector corresponds to a particular RTI call type and indicates the probability that a generated RTI call is of that type as well as the number of calls of that type generated. It follows from this definition that the elements of this vector should sum to 1.0. Only non-zero probabilities are maintained, to reduce memory requirements. If an RTI call type is selected based on this probability, the SIM component generates *n* calls of that type, where *n* is also specified in the vector. This is done to provide the capability of "bursty" call generation as would be required when instantiating a set of new objects. The third parameter is a *burst probability* that is used to control the generation of multiple RTI calls at a stretch. After generating an RTI call, the SIM component uses this probability to determine whether another call should be generated. If so, the *RTI call vector* is renormalized to eliminate calls that have already been generated during this execution of the object. Note, specifying a probability p is equivalent to making an average of $1/(1-p)^2$ calls.

	High	Medium	Quiescent
High	0.8	0.15	0.05
Medium	0.1	0.4	0.5
Quiescent	0.1	0	0.9

Table 2 - State Transition Matrix

In addition to these parameters, the simulation program computes and/or maintains other pieces of information for each object, including:

- an identifier
- pointers to access those federates that should receive updates and/or interactions generated by the object
- time since the last update was generated (for those objects that specify *minimum rate* attributes)
- other book-keeping information

Although class structures capture the hierarchy among object types, they do not capture the logical aggregations in military units (e.g. a brigade). To this end, the model allows the definition of *groups* of objects. Groups are specified at run-time through configuration files. Objects from different classes can be part of the same group. A group specifies all information related to activity state (i.e.

[†] The *publish/subscribe* mechanism of the HLA is used to specify the information flow between federates [DMSO96b].

initial state, transition matrix and selection vector) and RTI calls (Boolean flags, RTI call vector and burst probability). The main property of a group is that its constituent objects all use the parameters specified for the group and thus exhibit *identical* dynamics with respect to activity and RTI call generation. The rationale is that this capability allows a user to model an aggregate unit such as a platoon of tanks since the constituent tanks will likely act together.

To reduce the amount of information to be specified by the user, object parameters are specified at the class and group levels. Group parameters override those specified for the class - if an object belongs to a group, the group parameters are used; otherwise the class parameters are used. If necessary, future versions of the model will provide the capability to specify parameters for individual objects.

The HLA provides federates with the capability to express interest in only a subset of the attributes of another object, through the *subscribe* mechanism. In conjunction with the *publish* mechanism, the RTI uses this information to route messages appropriately[‡]. Since our design philosophy (no application semantics) precludes the modeling of the attributes of simulation objects, we have chosen to model attribute-based subscription using a probability parameter. Each subscription specifies a probability that may be thought of as a *level of subscription*. The SIM component uses this probability when an update is generated to decide whether a subscribing federate should receive that update or not. This construct may be used to model attribute-level subscription as follows. If federate A is interested in 40% of federate B's attributes and federate B updates only 60% of its attributes regularly of which 30% overlap with the 40% interest of federate A, then federate A should specify a subscription probability of 0.5 (30/60).

In our model, as in HLA federations, subscriptions are specified at the federate level, rather than at the object level. This has the desirable effect of reducing the amount of configuration information the user has to specify. We expect that each federate will include a *federate controller object* that is responsible for executing federate-level actions such as publishing/subscribing and instantiating/deleting new objects. The distinguishing feature of the federate controller object is that it will be the only object in the federate with a non-zero probability of generating these federate-level RTI calls.

3.3 RTI Cost Model

One of the goals of the model is to help evaluate alternate RTI implementations and/or design choices. As such, this requires that the model should be easily configurable to model different RTI designs. As noted earlier, we have taken an approach that separates the functional aspects of the RTI from the cost. The rationale behind this approach is that the functionality is going to be essentially the same across all implementations (dictated by the HLA Interface Specification [DMSO96b]), and different implementations can be characterized by the costs they impose on the federation.

At the time of writing, the RTI cost model is still being developed. For the present, we are focusing on object and declaration management services. With regard to these services, the RTI essentially acts as an information pipe. When updates and interactions are generated, they must

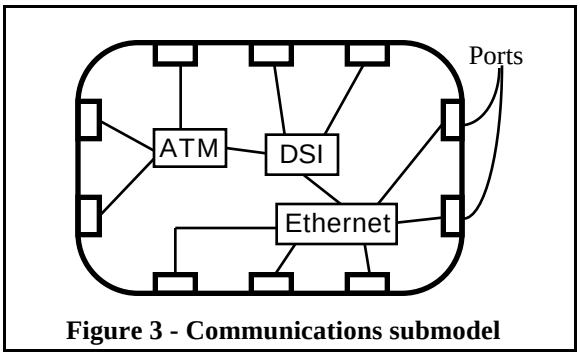


Figure 3 - Communications submodel

be transported through the RTI, which will incur the following potential costs:

- determination of receiver set - this could be as simple as a table lookup to determine a multicast group address
- generation and reception of physical messages - this will occur in the System component of the federate submodel
- any data distribution management overheads
- processing of received messages by the RTI at the receiver
- generation of corresponding calls to the federate by the RTI at the receiver

While this corresponds to the "push" model for data exchange, similar costs will be incurred for the "pull" model where a particular receiver initiates the data exchange.

Many RTI calls result in changes in the state of the RTI rather than data exchange between federates. This can involve the exchange of messages between the RTI components at various federates. The RTI cost model will provide the user with the capability to specify, for each type of RTI call, a sequence of message exchanges. Each

[‡] Data Distribution Management (DDM) builds upon the basic interest management of the publish/subscribe mechanism. Although DDM is not included in the simulation model at present, we are currently in the process of doing so.

step in this sequence could be: a unicast to a specific destination (such as a central RTI component), unicast to the sender (i.e. a reply), unicast to the originator of the sequence, multicast to a uniformly selected set of destinations or broadcast to all federates. Finally, users will be able to specify the local costs of all RTI call types (data exchange as well as control) using a general method that allows for constant values, distributions and general functions of other variables.

3.4 Time Management

The modeling of Time Management (TM) warrants some explanation. While performance in general is typically dependent on the federation and how it utilizes the various services, the performance of TM is perhaps most so. The reason is that TM deals with the advance of simulation time, which depends directly on the federates. A large part of TM is the *synchronization protocol* that provides time-stamp ordering. Performance analysis of these protocols is a well-established research area by itself [SrRe95]. Such detailed performance analyses are beyond the scope of this project. On the other hand, general cost estimates to fit the RTI cost model would result in over-simplification and therefore meaningless analyses. We are currently investigating some sort of middle ground. For the present, all aspects of TM are omitted from the model since we are focusing on protofederations that do not employ TM services.

3.5 Communications Submodel

The communications submodel encapsulates all communications in the federation. Figure 3 shows the structure of this submodel. Each federate connects to the submodel through a bi-directional *port*. The ports are interconnected by communication subsystems as shown. The submodel will consist of first-order models of various communication subsystems, which can be instantiated as desired to construct the topology. Communications are modeled at the message level. As messages are generated, they are routed through the topology using specified routes to their destinations. We expect to model multicast support in the communication subsystems, since multicast is likely to be the primary communication primitive. Currently, ATM is the only communications subsystem modeled. We expect to have DSI and Ethernet and possibly some constructs for modeling gateways and routers (potential bottlenecks) in the near future.

4. STATUS

The implementation of the simulation model is on-going. Validation of the model will be performed against data from the JPSD experiment, when that data becomes available. In its current state, which includes most of the functionality of the final version, the model runs quite efficiently - on the order of ten times faster than real time for a federation of 10 federates.

5. SUMMARY

We have described the design of a simulation model for first-order performance analyses of HLA-compliant federations. In order to be able to capture the wide variety of simulations (and federations) that the HLA will support, the model is extremely flexible. A consequence is that effort is required on the part of the user to configure the model to suit a particular federation. However, once this is done, a small set of parameters can be established as the control parameters across which the required analyses are performed. It is important to remember that this is a stochastic model - consequently, it is difficult and inappropriate to program deterministic scenarios. As with any simulation model, it is an approximation of reality, but one that we believe is powerful enough to provide useful insights into the design of future HLA federations. As simulation programs contemplate and move towards HLA compliance, this federation analysis tool can be used to support the transition process by demonstrating feasibility and alleviating concerns before bending metal, supporting design choices and identifying potential trouble-spots.

ACKNOWLEDGMENTS

The authors are grateful to Andy Gladyszak and Robin Arrieta of Scientific and Engineering Software, Inc., Austin, TX, for their patient support during the initial months of this contract. The simulation model is being implemented by Greg Borek of Mystech Associates, Inc. using the Workbench simulation tool. This work is sponsored by DMSO under contract number N61339-96-D-00020023 of the ADST II program.

REFERENCES

[DoD95] Department of Defense, Modeling and Simulation (M&S) Master Plan, October 1995.

[DMSO96a] Defense Modeling and Simulation Office, “HLA Glossary”, http://www.dmso.mil/hla/, August 1996.

[DMSO96b] Defense Modeling and Simulation Office, “HLA Interface Specification”, http://www.dmso.mil/hla/, August 1996.

[SrRe95] Srinivasan, S. and Reynolds, P.F. Jr., “NPSI Adaptive Synchronization Algorithms for PDES”, *Proceedings of the 1995 Winter Simulation Conference*, Dec. 1995, 658-665.

AUTHOR BIOGRAPHIES

SUDHIR SRINIVASAN is a research scientist at Mystech Associates, Inc., conducting research in parallel and distributed modeling and simulation. He received his Ph.D. in Computer Science from the University of Virginia in 1995 and Bachelor of Engineering also in Computer Science from the Bangalore University, India, in 1990. From August 1995 through February 1996, he was also a research associate at the University of Virginia, working on identifying fundamental issues in linking models at different levels of resolution. His main research interest is in parallel and distributed simulation, especially the High Level Architecture. His other interests are in parallel and distributed computing and networking. He has published several papers in conferences and journals and is a member of the ACM and the IEEE Computer Society.

PAUL F. REYNOLDS, JR., Ph.D., University of Texas at Austin, '79, is an Associate Professor of Computer Science at the University of Virginia. He has published widely in the area of parallel computation, specifically in parallel simulation, and parallel language and algorithm design. He has served on a number of national committees and advisory groups as an expert on parallel computation, and more specifically as an expert on parallel and distributed simulation. He has been a consultant to numerous corporations and government agencies in the systems and simulation areas. He is a member of the ACM and the IEEE Computer Society.